Лекция 8
Конвейерные вычислительные системы

Ефимов Александр Владимирович
E-mail: alexandr.v.efimov@sibguti.ru

Курс «Архитектура вычислительных систем»
СибГУТИ, 2020

Конвейерные ВС

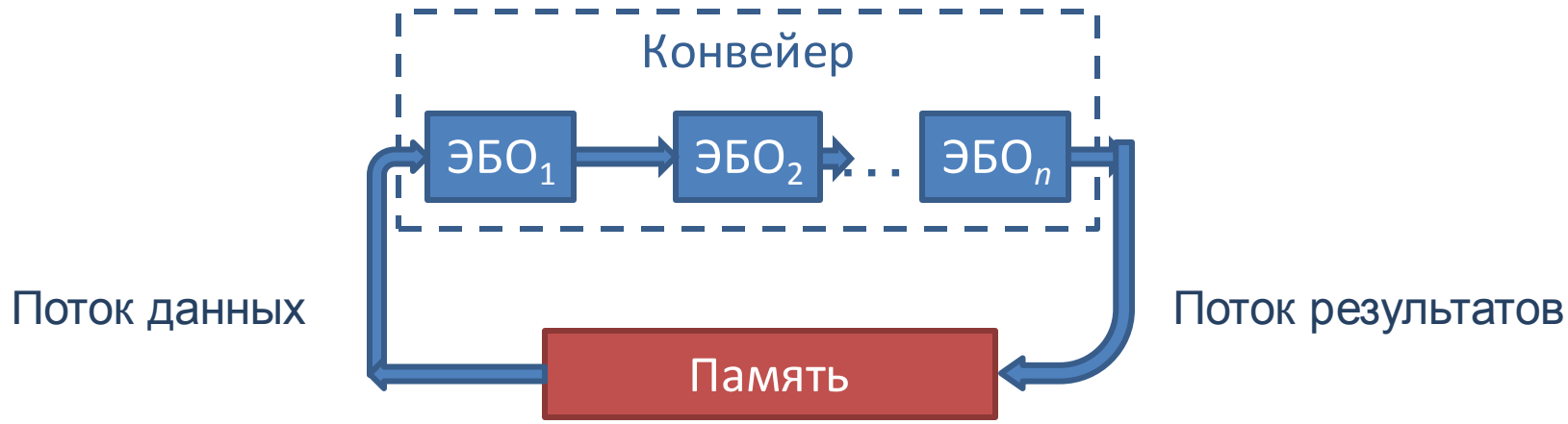
- Конвейерный способ обработки информации;
- аппаратурная реализация операции над векторами данных;

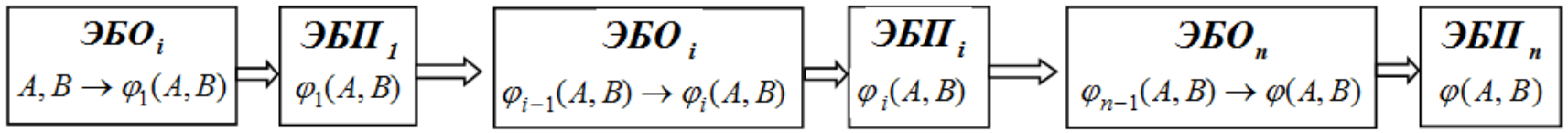
Например: $A+\alpha B$, $(A+\alpha)B$, где α - скаляр

$A = (A_1, A_2, \ldots, A_i, \ldots, A_n)$, $B = (B_1, B_2, \ldots, B_i, \ldots, B_n)$

- в 70-х и 80-х годах 20 века обеспечивали быстродействие порядка $10^8 - 10^9$ опер./с;
- современные быстродействующие микропроцессорные БИС основаны на конвейеризации вычислений.

Каноническая функциональная структура





$$A, B \to \varphi_1(A, B) \to \cdots \to \varphi_i(A, B) \to \cdots \to \varphi_n(A, B) = \varphi(A, B).$$

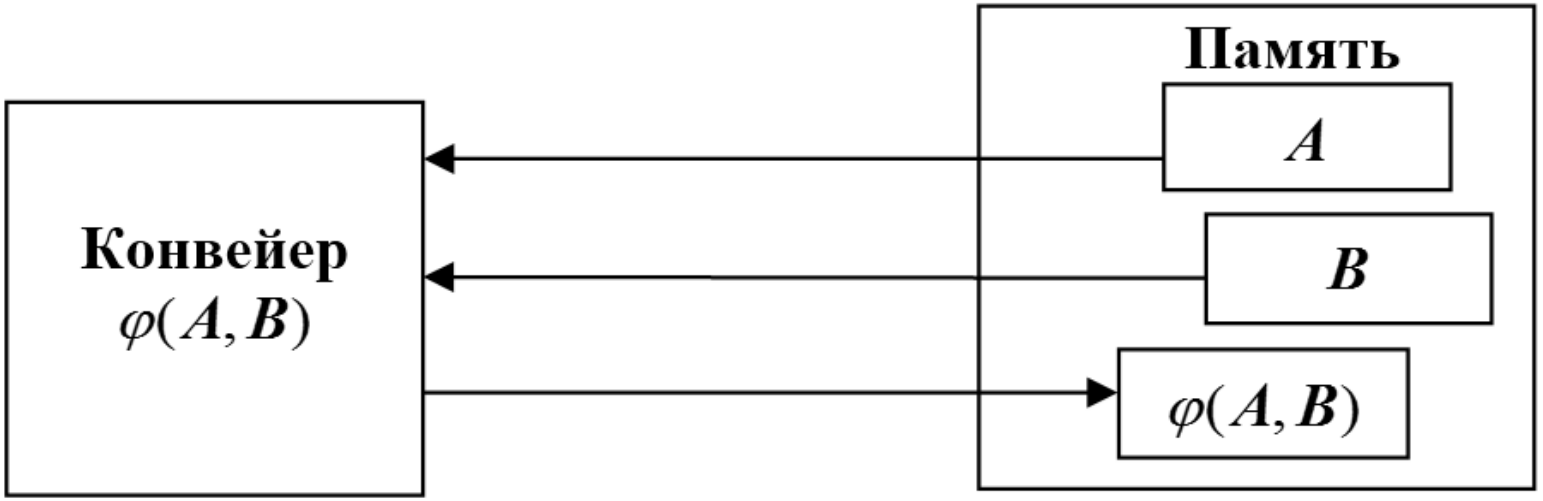
A, B – векторы-операнды; $\varphi_i(A, B)$ – частичное преобразование векторов A, B

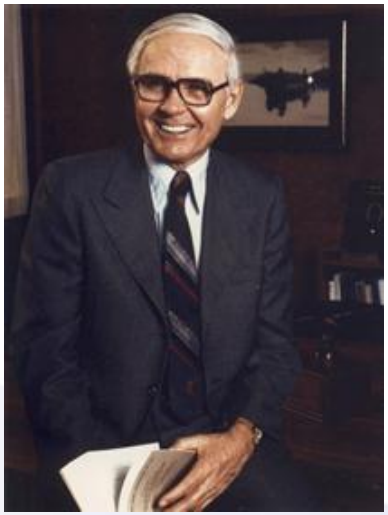
Функционирование конвейера (pipeline)

$A_1,B_1 \to \varphi_1(A_1,B_1)$				
$A_2,B_2 \to \varphi_1(A_2,B_2)$	…			
…	…			
$A_i,B_i \to \varphi_1(A_i,B_i)$	…	$\varphi_{i-1}(A_1,B_1) \to \varphi_i(A_1,B_1)$		
$A_{i+1},B_{i+1} \to \varphi_1(A_{i+1},B_{i+1})$	…	$\varphi_{i-1}(A_2,B_2) \to \varphi_i(A_2,B_2)$	…	
	…	…	…	
$A_n,B_n \to \varphi_1(A_n,B_n)$	…	$\varphi_{i-1}(A_{n-i+1},B_{n-i+1}) \to$ $\varphi_i(A_{n-i+1},B_{n-i+1})$	…	$\varphi_{n-1}(A_1,B_1) \to \varphi(A_1,B_1)$
$A_{n+1},B_{n+1} \to \varphi_1(A_{n+1},B_{n+1})$	…	$\varphi_{i-1}(A_{n-i+2},B_{n-i+2}) \to$ $\varphi_i(A_{n-i+2},B_{n-i+2})$	…	$\varphi_{n-1}(A_2,B_2) \to \varphi(A_2,B_2)$
…	…	…	…	…

Конвейерные систем типа «память-память»

CDC (Control Data Corporation), Сеймур Крей и Уильям Норрис

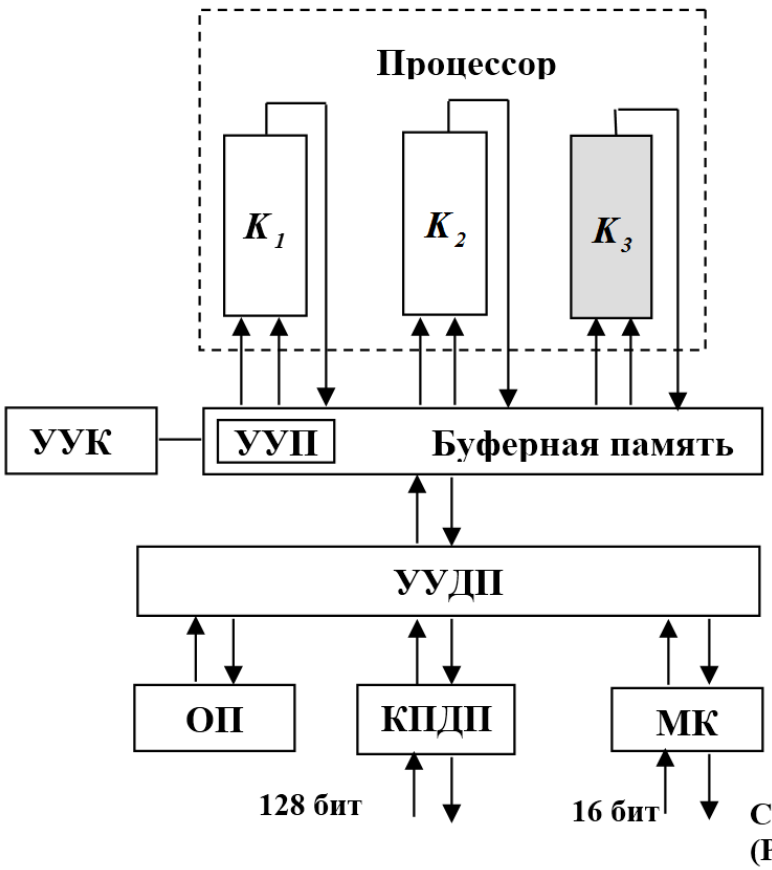




Вычислительная система STAR-100

- STAR – STring ARray computer – векторный компьютер;
- разработка с 1965 по 1973;
- быстродействие – 10^8 оп./сек.;
- стоимость 15 млн. $;
- создавалась под непосредственным влиянием языка программирования APL (A Programming Language): Язык APL (или АПЛ) – диалоговый язык программирования, характеризуется развитыми средствами работы с регулярными структурами данных (векторами, матрицами, массивами) и богатым набором базовых операций и компактностью записи.

Функциональная структура STAR-100



$К_1$, $К_2$ – конвейеры выполнения векторных операций (floating-point pair pipelines);
$К_3$ - конвейер реализации операций над калярными операндами (string data pipeline);
УУК - устройства управления командами;
УУП - устройства управления потоками;
УУДП - устройства управления доступом к памяти;
КПДП - канала прямого доступа в память;

МК – мультиплексированный канал;

Функциональные блоки STAR-100

- Каждый конвейер $К_1$, $К_2$, $К_3$ мог включать **до 30 блоков** обработки информации;
- Конвейеры $К_1$, $К_2$, $К_3$ имели **программируемую структуру**, следовательно на одном и том же множестве элементарных блоков обработки можно выполнять различные арифметические операции;
- В конвейерах $К_1$ и $К_2$ использовался служебный булевский вектор для обеспечена **избирательная обработка компонентов** векторов-операндов;
- Состав элементарных блоков обработки информации конвейеров выбран с учетом распределения вероятностей использования микроопераций различных типов;
- Каждый конвейер воспринимал **64-разрядный код** либо как один 64-разрядный операнд, либо как два 32-разрядных операнда.
- Время выполнения операции над парой операндов в любом из блоков конвейеров не превышало 40 нс.

Функциональные блоки STAR-100

УУК имело буфер опережающего просмотра команд (емкостью в четыре 512-разрядных суперслова) со стековым механизмом работы;

УУП использовалось для управления потоками операндов и команд между УУДП, конвейерами и УУК;

УУДП управляло 4-мя параллельными каналами обращения к памяти (2 – чтение операндов в конвейеры $К_1$ и $К_2$, 1 – запись результатов, 1 – работа с устройствами ввода-вывода);

Буферная память совокупность регистров с временем цикла 40 нс;

ОП (оперативная память) использовалась для хранения программ и данных. Реализована на магнитных сердечниках. Ёмкость 512 – 1024 К 64-разрядных слов, т.е. до 8 Мбайт. Время цикла памяти равно 1,28 мкс.

Функционирование STAR-100

Набор команд STAR-100 из 230 шт.:
65 команд предназначалось для работы с векторами данных;
130 команд для работы со скалярами;

Операционная система (ОС) STAR-100 относилась к классу распределенных;
функции реализовывались специальной вычислительной сетью из 9 мини-машин;

Система программирования STAR-100 включала компиляторы с языков APL-STAR, COBOL и FORTRAN.

Конвейерные систем типа «регистр-регистр»

Cray Research Inc

Сеймур Крей

CRAY INC.

Вычислительная система Cray-1

Cray Research Inc, 1976 г.

Технические характеристики:

- быстродействие – 160 MFLOPS ($16*10^7$опер./с при выполнении операций с плавающей запятой над векторами данных и 37 млн. опер./с – над скалярами);
- емкость оперативной памяти 8–64 М байт;
- длина слова данных – 64 двоичных разряда;
- цена – 5 – 9 млн.$.

Функциональная структура Cray-1

Функциональные блоки Cray-1

Подсистема управления программой состояла из стандартных устройств (счетчик команд, средства организации ветвлений, устройство прерывания и т.п.) и буферной памяти для команд.

Подсистема конвейеров - композиция из 4 групп функционально ориентированных конвейеров:

- для операций над адресами,
- для скалярных операций,
- для операций над числами с плавающей запятой,
- для векторных операций.

Всего 12 конвейеров,

длительность цикла любого ЭБО – 12,5 нс ($12,5*10^{-9}$ с).

Функциональные блоки Cray-1

Группа конвейеров для операций над адресами:

- конвейер для сложения целых чисел (2 ЭБО);
- конвейер для умножения целых чисел (6 ЭБО).

Группа конвейеров для скалярных операций:

- конвейер счетчик (3 ЭБО);
- конвейер сложения целых чисел (3 ЭБО);
- конвейер логических операций (1 ЭБО);
- конвейер сдвига (3 ЭБО).

Конвейер для операций над числами с плавающей запятойб

Конвейер для векторных операций.

Функциональные блоки Cray-1

Конвейер операций над числами с плавающей запятой:

- конвейер сложения (6 ЭБО);
- конвейер умножения (7 ЭБО);
- конвейер вычисления обратной величины (14 ЭБО).

Конвейер для векторных операций:

- конвейер сложения целых чисел (3 ЭБО);
- конвейер логических операций (2 ЭБО);
- конвейер сдвига (4 ЭБО).

Функциональные блоки Cray-1

Подсистема регистров включала регистры с программным доступом:

1) восемь 24-разрядных адресных *A*-регистров;

2) шестьдесят четыре 24-разрядных промежуточных адресных *B*-регистров;

3) восемь 64-разрядных скалярных *S*-регистров;

4) шестьдесят четыре 64-разрядных промежуточных скалярных *T*-регистров;

5) восемь векторных *V*-регистров, каждый из которых способен хранить вектор из 64-х 64-разрядных компонентов.

Функциональные блоки Cray-1

Подсистема регистров по сути сверхоперативная память (с циклом 6 нс), обладающая емкостью 4888 байт, включала регистры с программным доступом:

1) восемь 24-разрядных адресных *A*-регистров;

2) шестьдесят четыре 24-разрядных промежуточных адресных *B*-регистров;

3) восемь 64-разрядных скалярных *S*-регистров;

4) шестьдесят четыре 64-разрядных промежуточных скалярных *T*-регистров;

5) восемь векторных *V*-регистров, каждый из которых способен хранить вектор из 64-х 64-разрядных компонентов.

Функциональные блоки Cray-1

Подсистема памяти и ввода-вывода

- оперативная память емкостью 1 М слов и состояла из 16 независимых банков по 64 К слов на 72 модулях памяти;
- время цикла одного банка 50 нс (4 цикла системы).

Подсистема ввода-вывода

- 12 входных и 12 выходных каналов, которые обеспечивали суммарную скорость 500 тыс. 64-разрядных слов в секунду.

Система команд Cray-1

128 основных команд, которые могли иметь одну или две 16-разрядных части:

- в виде одной части 7 разрядов отводилось под код операции, по 3 разряда – для адресов двух регистров, в которых хранились два операнда, 3 разряда – для адреса регистра, в который заносился результат;
- в виде двух частей использовались и для адресации основной оперативной памяти.

Числа с фиксированной запятой - 24 или 64 разряда.

Числа с плавающей запятой – 49 мантисса, 15 – порядок.

Операционная система Cray-1

Cray Operating System (COS) обеспечивала режим пакетной обработки (до 63 задач), а так же включала:

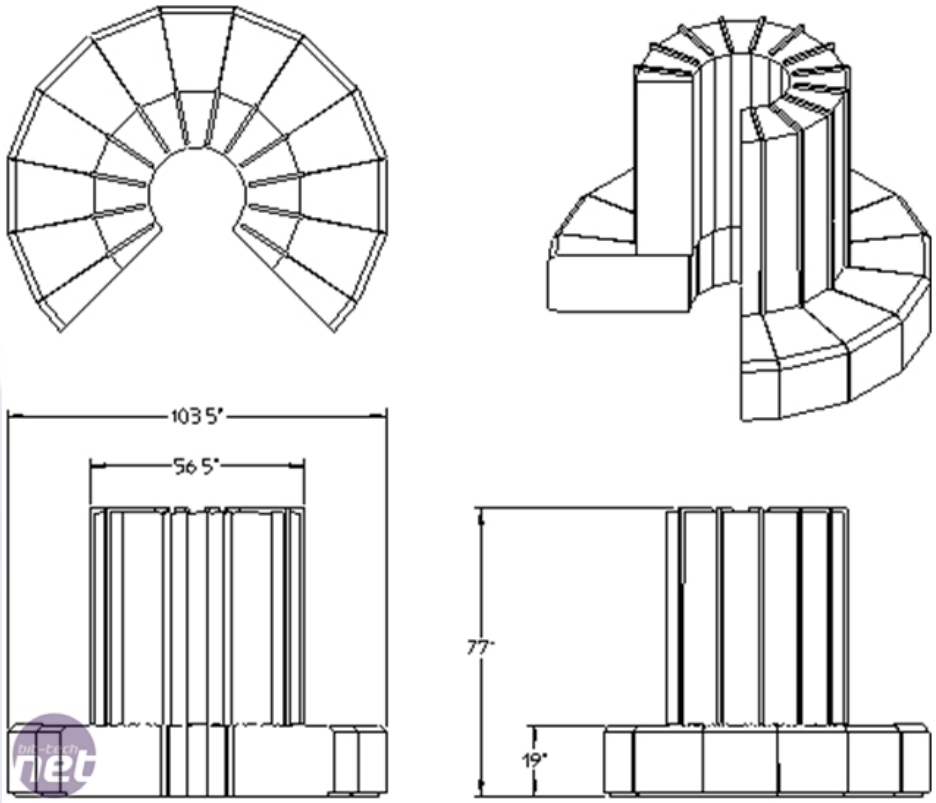
- оптимизирующий транслятор CFT (Cray Fortran Compiler) для языка высокого уровня ANSI 66 FORTRAN IV;
- макроассемблер CAL (Cray Assembler Language);
- библиотека стандартных программ;
- загрузчик и другие сервисные средства.

Особенности архитектуры Cray-1

Обладала способностью **адаптации к структуре решаемой задачи** за счет программного формирования цепочек (макроконвейеров) из произвольного числа конвейеров и с произвольной их последовательностью.

В системе допускалась **параллельная работа как конвейеров, так и элементарных блоков обработки** в пределах любого конвейера.

Конструкция Cray-1



1 фут = 0,3048 м

Конвейер команд

- Конвейерная обработка инструкций – это метод реализации CPU, при котором множество операций над несколькими инструкциями перекрываются
  - Конвейерная обработка инструкций использует программный параллелизм уровня инструкций (Instruction-Level Parallelism, ILP)
- Конвейеризация увеличивает пропускную способность CPU – среднее число инструкций, завершенных за такт
  - В идеальном случае происходит завершение одной инструкции за машинный такт
- Конвейеризация не сокращает время выполнения отдельной инструкции (также называемое временем задержки завершения инструкции, латентность)
  - Минимальное время задержки завершения инструкции - n тактов, где n – число ступеней конвейера
- Конвейер, описанный здесь, называется **упорядоченным** (**in-order**) **конвейером**, так как инструкции обрабатываются или исполняются в порядке, указанном в исходной программе

Функционирование конвейера (pipeline)

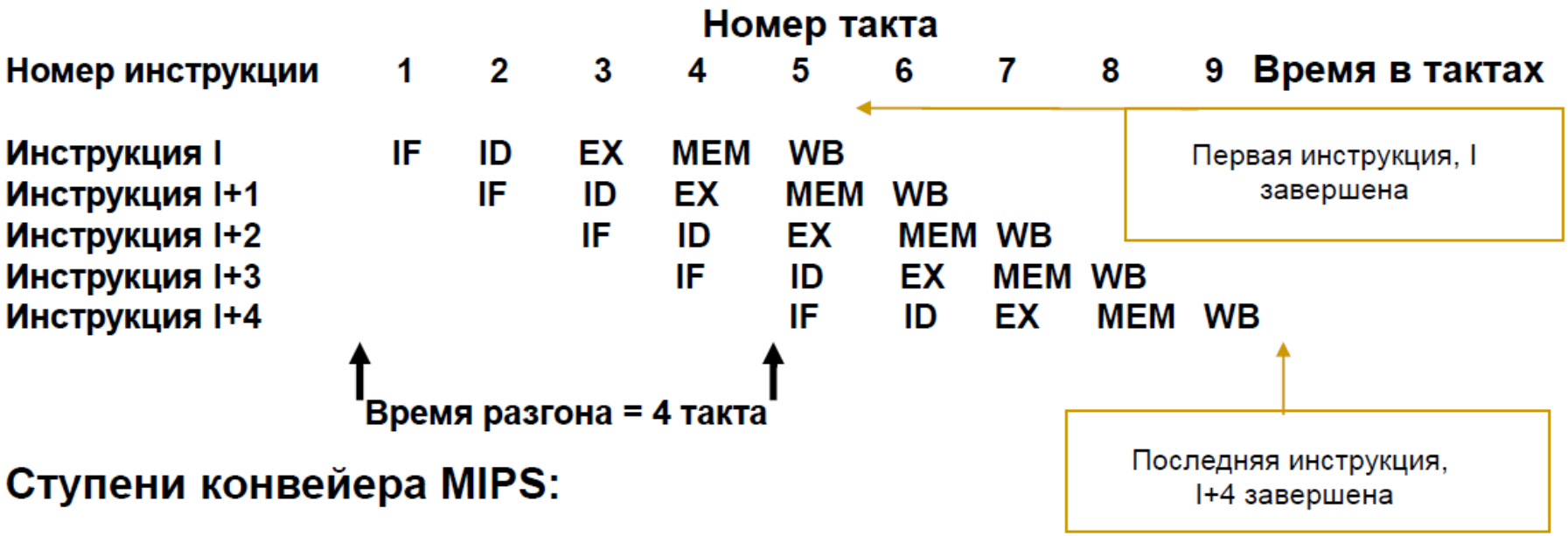
1958 г. ЭВМ М-20 реализовано совмещение IF и EX

1963 г. ЭВМ ATLAS реализовано совмещение IF, ID, RD и EX

1966 г. ЭВМ БЭСМ-6 реализован конвейерный процессор

Конвейер с упорядоченной обработкой целочисленных операций

Число тактов до заполнения = время разгона = число ступеней - 1



Ступени конвейера MIPS:

IF = Выборка инструкции (Instruction Fetch)

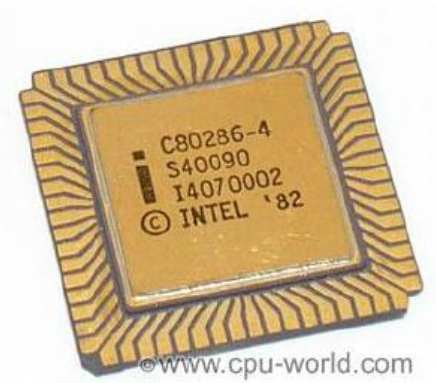
ID = Декодирование инструкции (Instruction Decode)

EX = Исполнение (Execution)

MEM = Обращение к памяти (Memory Access)

WB = Запись результата (Write Back)

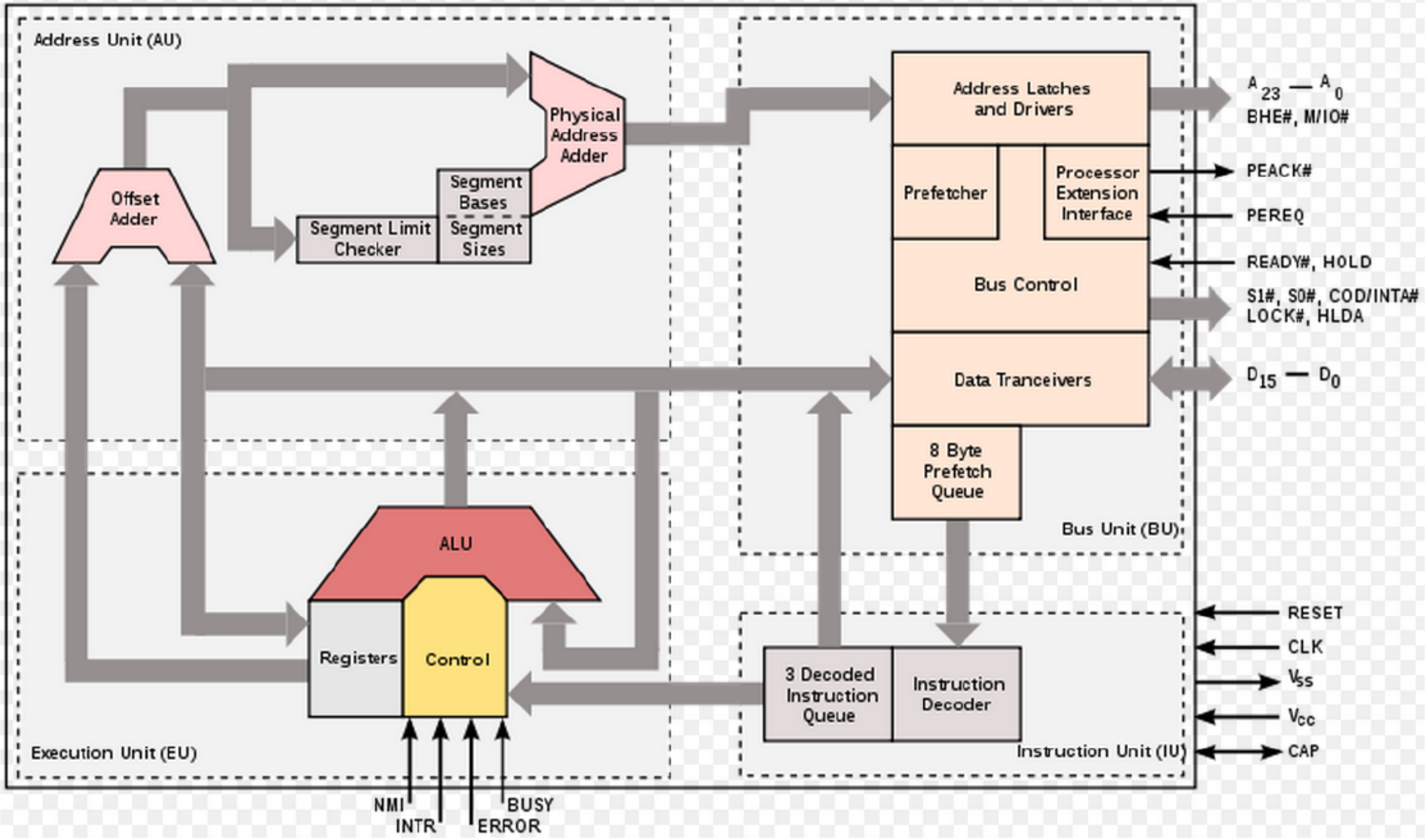
Intel 80286



- Intel 80286 (1982)
  - Тактовая частота – 6 (12,5) МГц
  - Производительность – 0,9 (2,66) MIPS
  - Количество транзисторов – 134 000
  - Площадь кристалла – 49 кв.мм
  - Техпроцесс – 1,5 мкм
  - 0.21 Instructions Per Clock
  - Конвейер команд (длина - 4)
  - Защищенный режим
  - Linear Memory Management Unit (MMU)
- Intel 80287 (1983)
  - Математический сопроцессор
  - Производительность ~65 000 FLOPS

Pentium 4 – 31
Core 2 – 14
Sandy Bridge – 18
Ivy Bridge – 14
Haswell, Broadwell,
SkyLake, Kaby Lake –14-19

Intel 80286



http://en.wikipedia.org/wiki/File:Intel_i80286_arch.svg

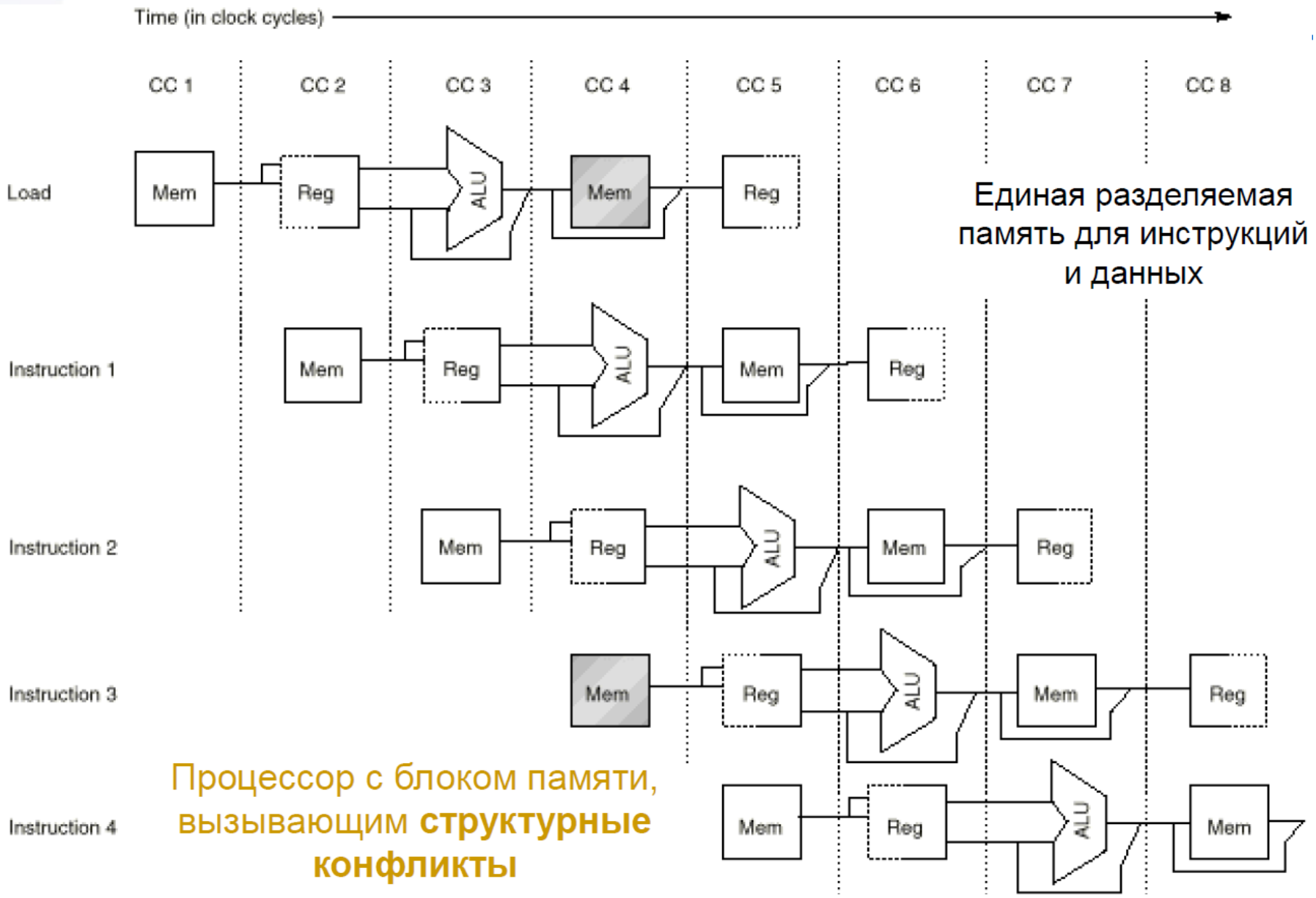
Пример

- При идеальном выполнении программы на 5-ступенчатом конвейере она выполнится за (4+16*7)+4 = 120 тактов (ранее – 580)

```
MOV   F1, 0.0
MOV   R1, 0
MOV   R2, 16
LOAD  F2, [R3+R1]
LOAD  F3, [R4+R1]
DMUL  F2, F2, F3
DADD  F1, F1, F2
INC   R1
CMP   R1, R2
JL    -6
ST    F1, (C)
```

Конфликты конвейера команд

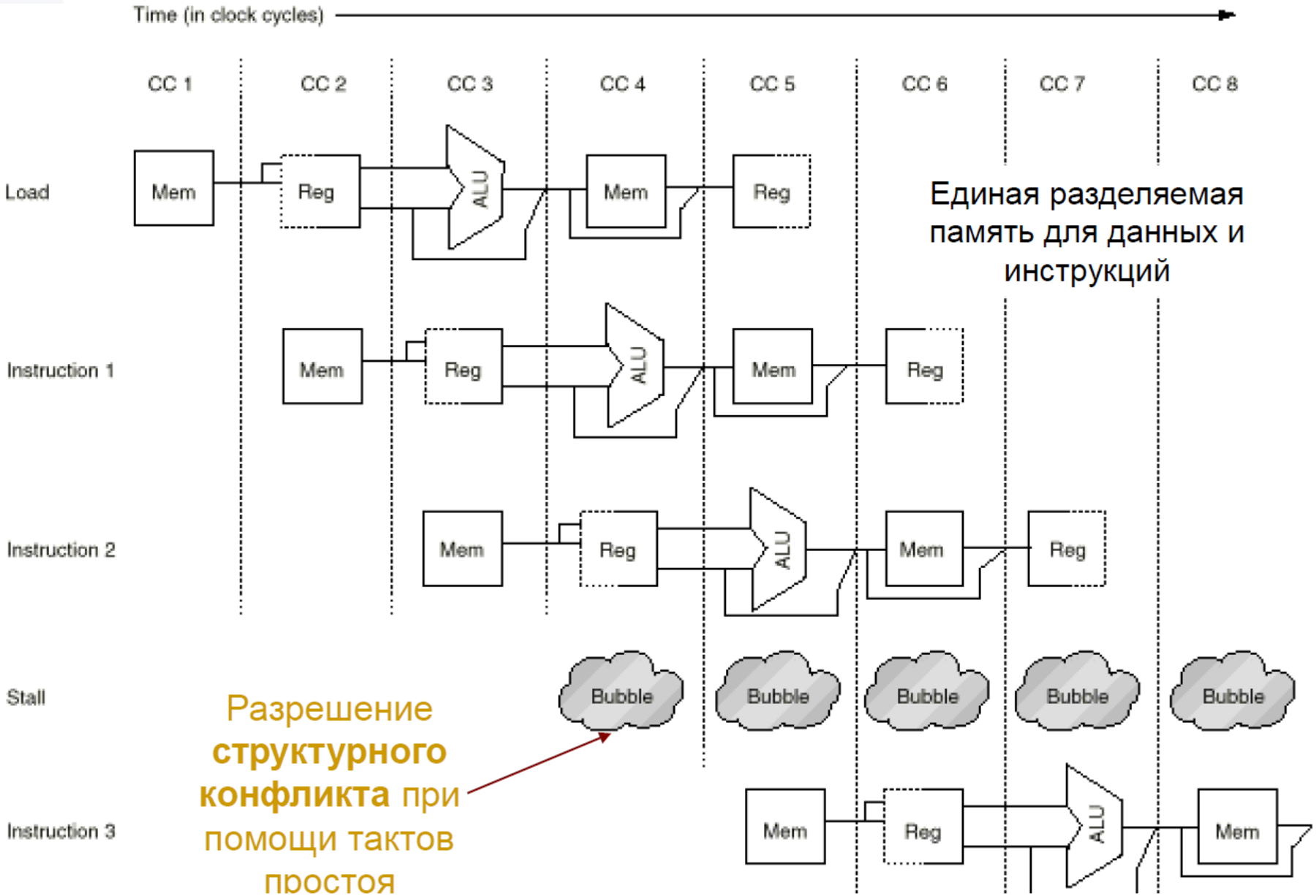
- Структурные конфликты
  - Возникают из-за недостатков аппаратных ресурсов, когда доступное аппаратное обеспечение не в состоянии поддерживать все возможные комбинации параллельного исполнения инструкций
- Конфликты данных
  - Возникают когда инструкция зависит от результата выполнения предыдущей инструкции так, что это проявляется при перекрытии инструкций в конвейере
- Конфликты управления
  - Возникают при конвейеризации условных переходов и других инструкций, которые изменяют PC (program counter)



Единая разделяемая память для инструкций и данных

Процессор с блоком памяти, вызывающим **структурные конфликты**

В машине с единственным портом памяти будет возникать конфликт при любом обращении к памяти

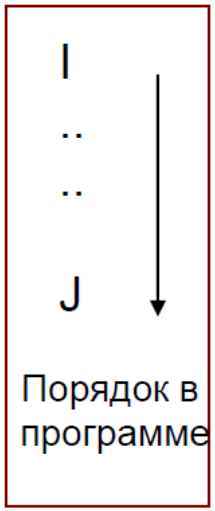


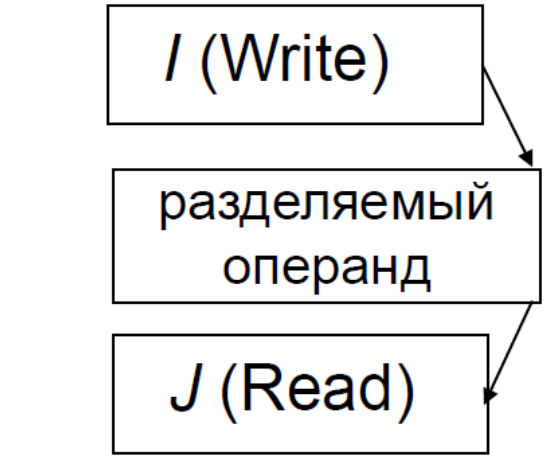
Единая разделяемая память для данных и инструкций

Разрешение **структурного конфликта** при помощи тактов простоя

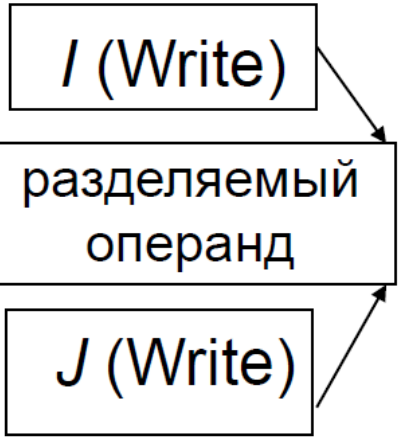
Структурный конфликт приводит к необходимости вставки «пузырей» в конвейер.

Конфликты данных

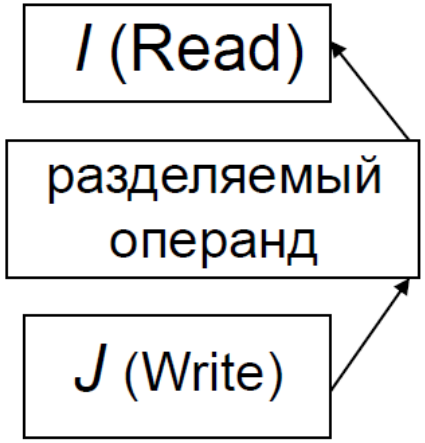




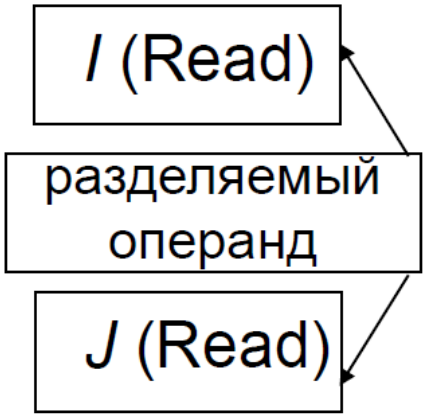
Read after Write (RAW)



Write after Write (WAW)



Write after Read (WAR)

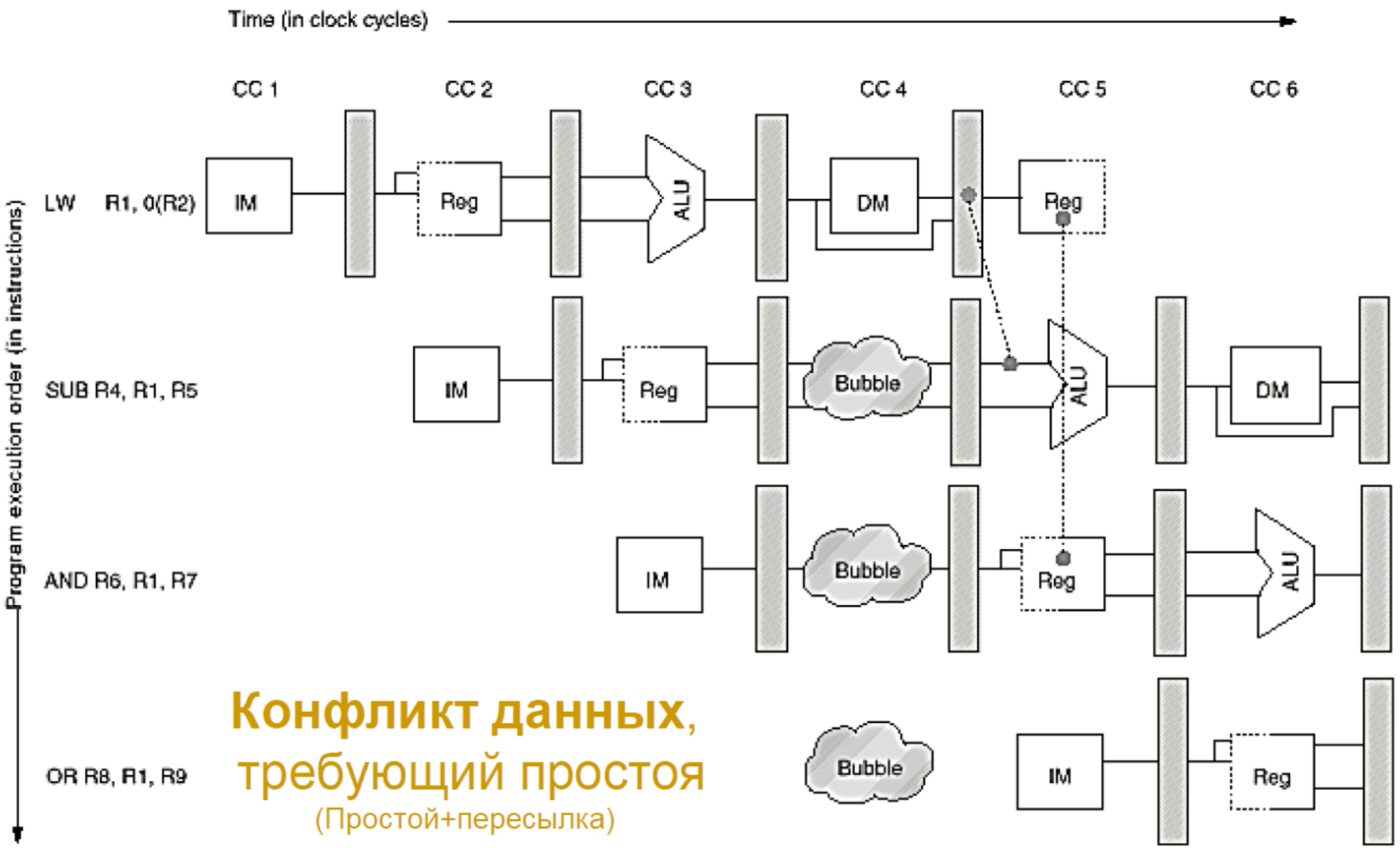


Read after Read (RAR)

нет конфликта

Пример **конфликта данных**

Использование результата инструкции DADD в инструкциях 2-4 вызывает конфликт данных: пока не выполнится запись в регистр, эти инструкции не смогут его прочитать.



Конфликт данных, требующий простоя

(Простой+пересылка)

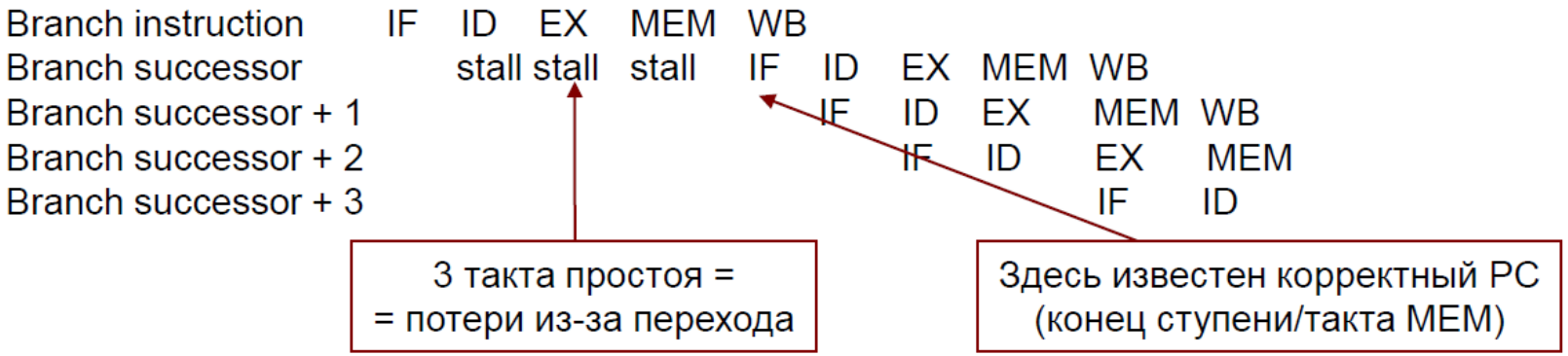
Блокировка при загрузке приводит к необходимости вставки «пузыря» на такте 4, задерживая инструкцию SUB и следующие за ней на 1 такт

Конфликт управления

При выполнении условного перехода, может измениться PC что, если не принять специальных мер, ведет к остановке конвейера на много тактов, пока не будет вычислено условие перехода (переход определится).

- Иначе PC может быть неверным, когда потребуется на ступени IF.

В рассматриваемом конвейере MIPS, условный переход определится на ступени 4 (MEM), приводя к трем тактам простоя, как показано ниже:



Предположим, что мы останавливаем или сбрасываем конвейер при инструкции перехода, тогда в рассматриваемом конвейере MIPS, тратится впустую три такта для каждого перехода.

Потери из-за перехода = номер ступени, когда переход определится - 1

здесь потери = 4 - 1 = 3 такта

Статическое предсказание переходов

- Статическое предсказание переходов кодируется в инструкциях перехода, используя один бит предсказания:
  0 = не происходит, 1 = происходит
  - Требует поддержки ISA
- Существует два основных метода для статического предсказания переходов во время компиляции:
  - Сбор информации о поведении программы при ее запусках и использование при перекомпиляции (профилирование)
    - Например, профиль программы может показать, что большинство переходов вперед и назад (это часто вызвано циклами) происходят. Простейшая схема в данном случае - всегда предсказывать, что переход происходит
- Эвристическое предсказание переходов на основе направления перехода, помечая переходы назад как происходящие и переходы вперед как не происходящие

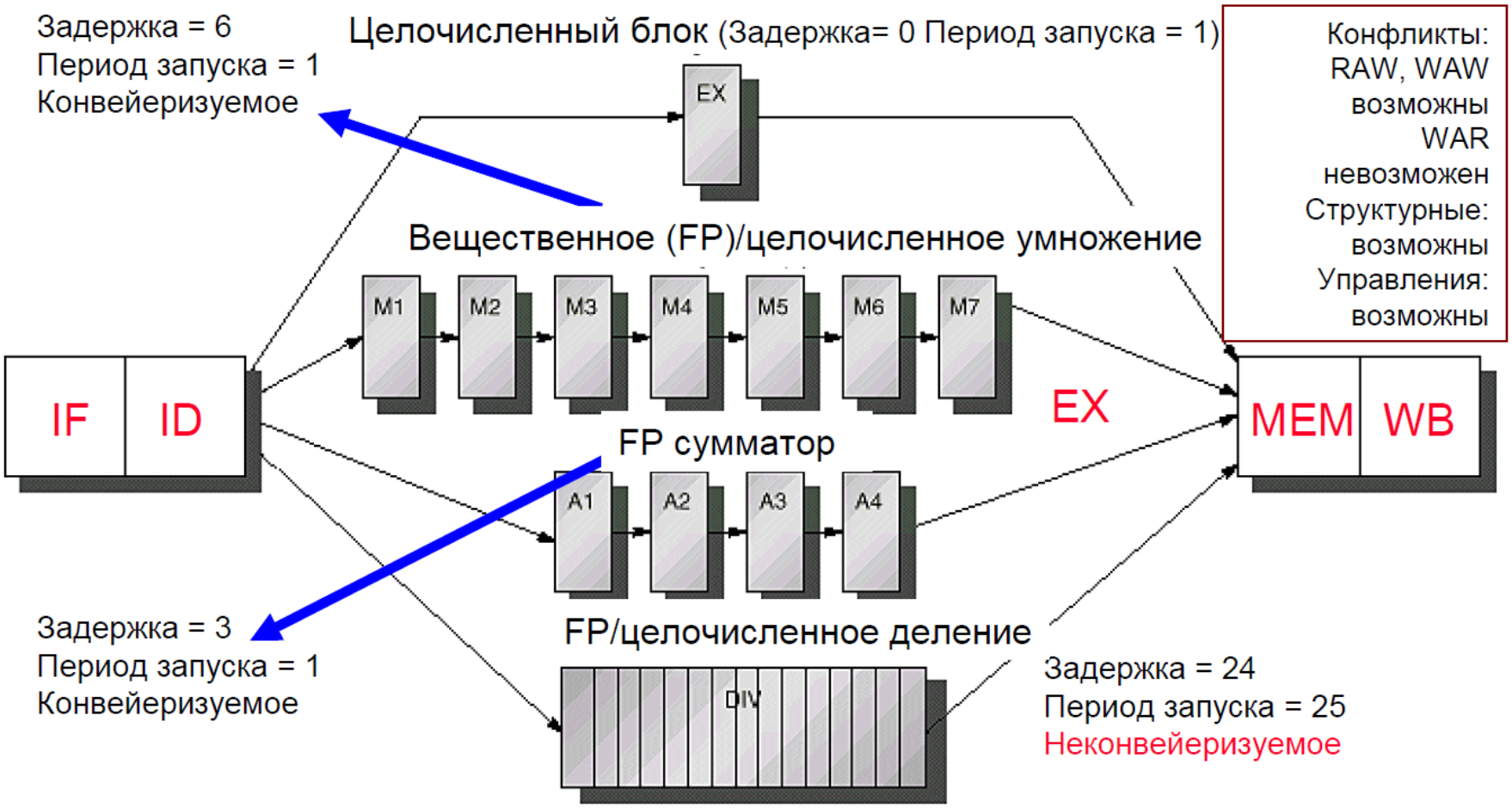
Динамическое предсказание переходов

- Предположение о направлении перехода основывается на истории переходов
- Пример: двухуровневый предсказатель с глобальной историей
  - Хранит результаты для M последних использованных инструкций перехода
  - Для каждой хранятся последние N переходов
  - Sandy Bridge использует 32-битный регистр для хранения истории переходов
    - Точность предсказания >90%
- Процессор загружает на конвейер инструкции из предсказанной ветки перехода, в случае неверного предсказания результат их исполнения не используется

Рекомендации

- Размещать наиболее вероятные ветви в начале ветвлений
- Выносить выше (по уровню вложенности в циклах) инвариантные ветвления
- Использовать разворачивание циклов

Многотактовый конвейер вещественных операций



Однопортовый конвейер MIPS с упорядоченной обработкой и поддержкой FP
Супер-конвейерный CPU: Конвейерный CPU с конвейеризуемыми FP блоками

Задержки инструкций

DisplayFamily_DisplayModel	Latency 06_3AH	Latency 06_2AH, 06_2DH	Throughput 06_3AH	Throughput 06_2AH, 06_2DH
VMOVDDUP ymm1, ymm2		1		1
VMULPD/PS ymm1, ymm2, ymm3		5		1
VSUBPD/PS ymm1, ymm2, imm		3		1
VDIVPD ymm1, ymm2, ymm3	35	45	28	44
VDIVPS ymm1, ymm2, ymm3	21	29	14	28
VSQRTPD ymm1, ymm2	35	45	28	44
VMULPD/PS ymm1, ymm2, ymm3		5		1
VRSQRTPS ymm1, ymm2		7		1
FSQRT EP		43		X87 FPU
F2XM1		90-150		X87 FPU
FCOS		190-240		X87 FPU
FPATAN		150-300		X87 FPU

Simultaneous Multi-Threading
Одновременная многопоточность

- При выполнении большинства операций оказываются полностью задействованными не все составные компоненты процессоров
- При использовании одновременной многопоточности на одном конвейере выполняются несколько программных потоков
  - Процессор дополняется средствами запоминания состояния потоков, схемами контроля одновременного выполнения нескольких потоков и т. д.
  - Одновременно выполняемые потоки конкурируют за исполнительные блоки единственного процессора, что приводит к возникновению структурных конфликтов

Суперскалярность

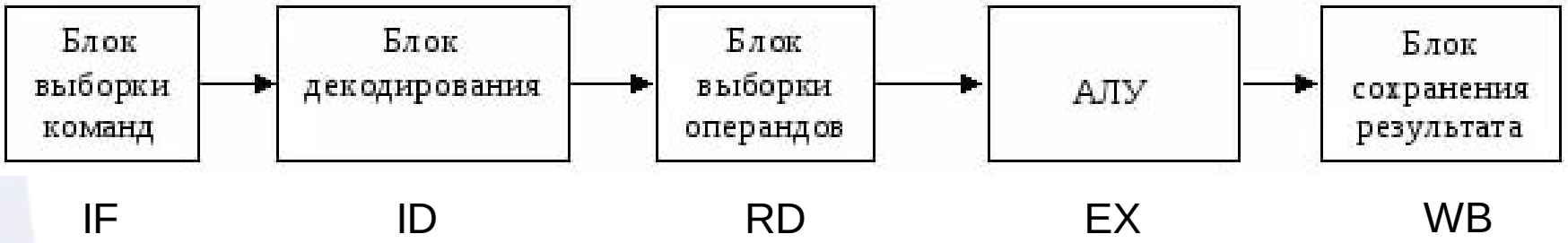
- Если инструкции, обрабатываемые конвейером, не противоречат друг другу, и ни одна не зависит от результата другой, то они могут быть выполнены параллельно
- Суперскалярность — запуск на выполнение нескольких инструкций за один такт для использования нескольких исполнительных блоков на различных стадиях конвейера
  - Используются несколько декодеров инструкций
  - Планирование исполнения потока инструкций является динамическим
  - Основывается либо на анализе зависимостей между инструкциями, либо на наборе правил

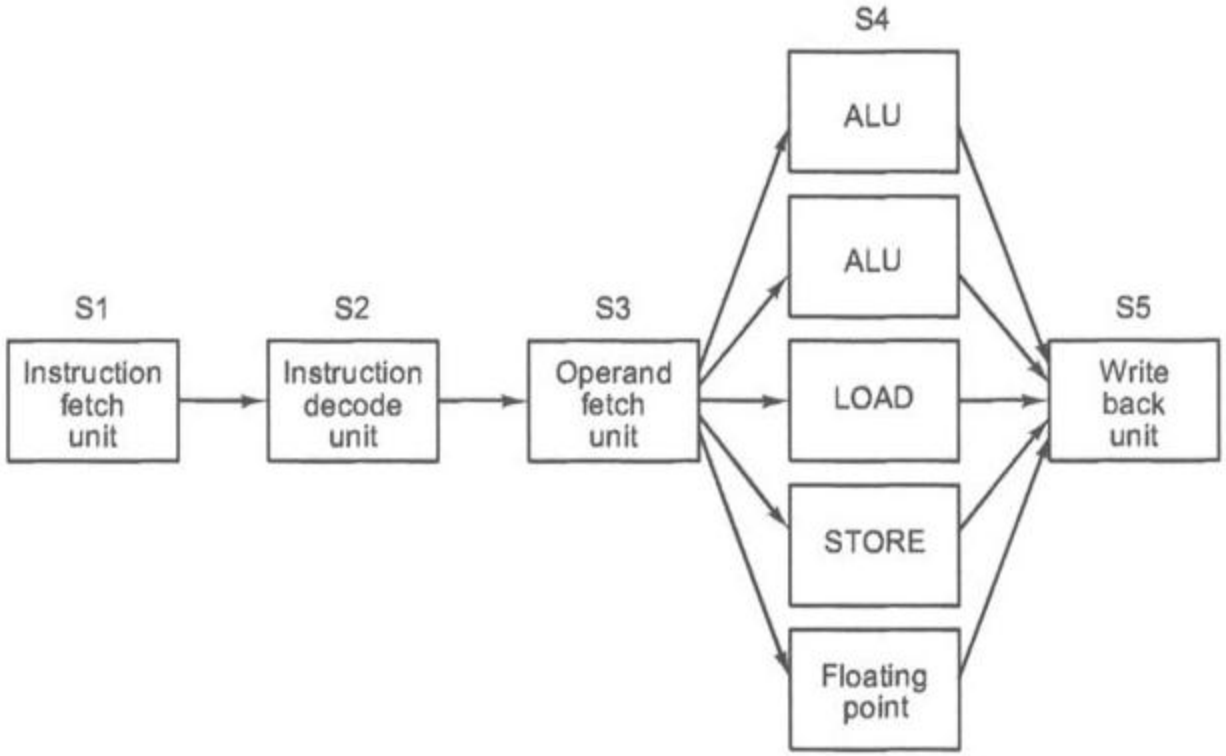
Intel Pentium

- Intel Pentium (1993)
  - Тактовая частота – 60 (233) МГц
  - Производительность – 120 (400) MIPS
  - Количество транзисторов – 3,1 (3,3) млн.
  - Площадь кристалла – 294 (90 и 83) кв.мм
  - Техпроцесс – 0,8 (0,35) мкм
  - Длина конвейера – 5
  - Суперскалярная архитектура
  - Механизм предсказания адресов ветвления
  - Встроенный математический сопроцессор
    - Производительность – ~1,44 MFLOPS
  - Кеш
    - L1: 16 Кбайт (8 Kb Data + 8 Kb Code)
    - L2: на материнской плате 1 Мбайт
    - 4-канальная наборно-ассоциативная архитектура
  - Тепловыделение 8 (15) Вт
    - Требуется внешнее охлаждение

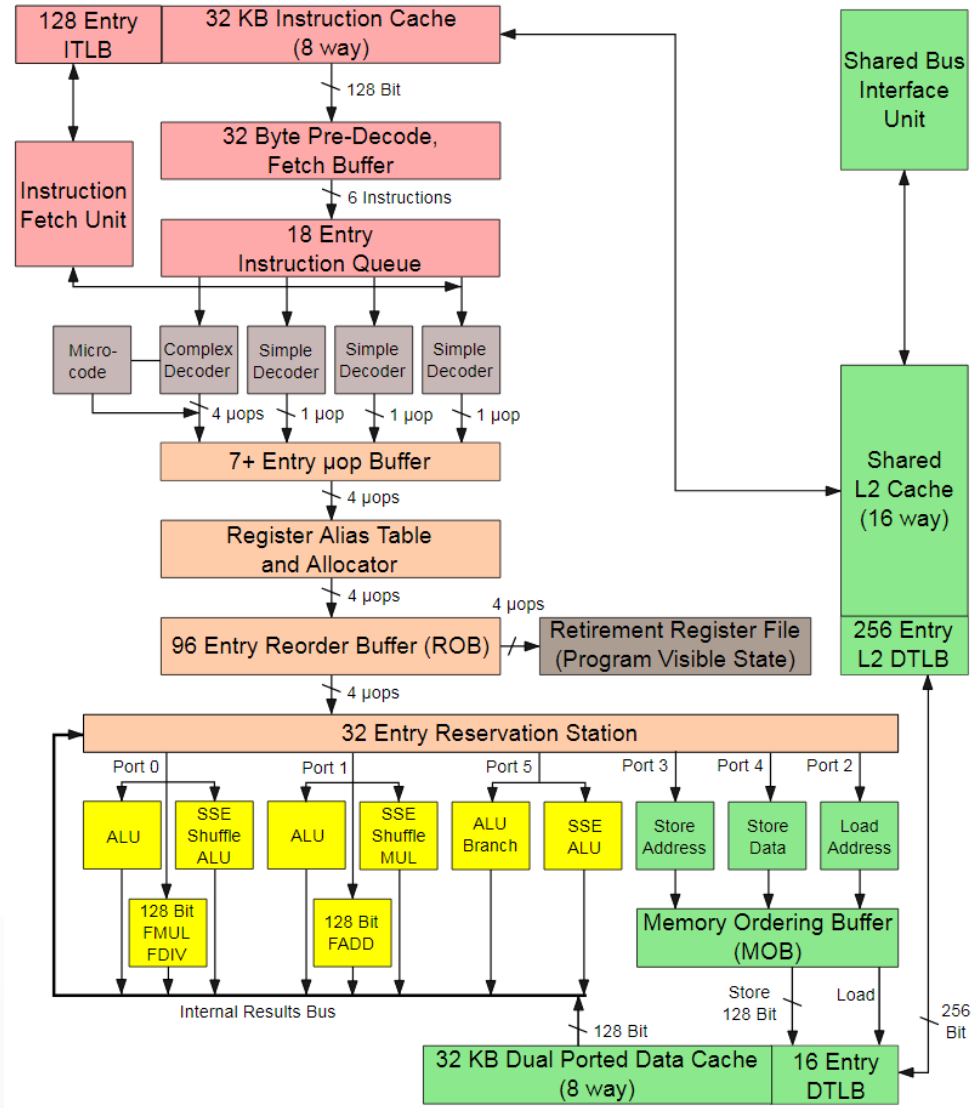
Рисунок: http://commons.wikimedia.org/wiki/File:Pentium-60-front.jpg?uselang=ru

Функциональная структура суперскалярного процессора



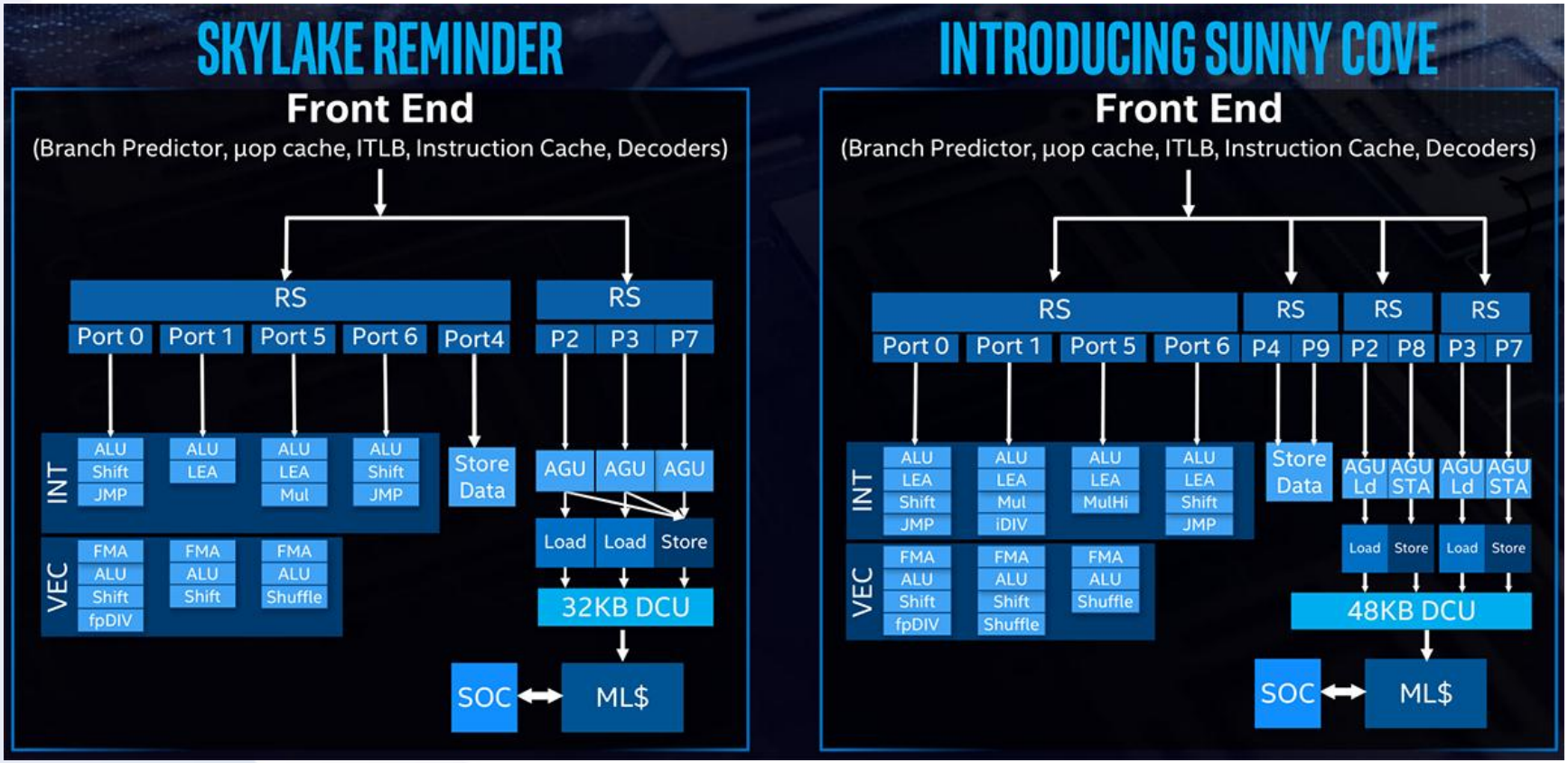


Пример функциональной структуры суперскалярного процессора



Intel Core 2 Architecture

Пример функциональной структуры суперскалярного процессора



Литература

Хорошевский В.Г. Архитектура вычислительных систем. Учебное пособие. – М.: МГТУ им. Н.Э. Баумана, 2005; 2-е издание, 2008.

Хорошевский В.Г. Инженерные анализ функционирования вычислительных машин и систем. – М.: “Радио и связь”, 1987.